

# Type-safe Web APIs with Protocol Buffers in Swift

A blurred background image of a person with dark hair, wearing a patterned shirt, looking down at a screen.

Hi, I'm Yusuke  
@kitasuke

# I'm going to talk about...

→ What Protocol Buffers are

# I'm going to talk about...

- What Protocol Buffers are
- Pros and Cons

# I'm going to talk about...

- What Protocol Buffers are
- Pros and Cons
- Where to start

# Web APIs



**HTTP Request**  
**JSON** ➔ **Data**

```
let json = ["id": 1]
let data = try? JSONSerialization.data(withJSONObject: json,
options: .prettyPrinted)

var request = URLRequest(url: url)
request.httpBody = data
```

# HTTP Response

Data → JSON

```
// {"user": {"name": "Yusuke"}}

URLSession.shared.dataTask(with: request) { (data, _, _) in
    let json = (try? JSONSerialization.jsonObject(with: data!, options: .allowFragments)) as? [AnyHashable: Any]
    let user = json?["user"] as? [AnyHashable: Any]
    let name = user?["name"] as? String
}
```

```
{"isTyped": false,  
"status": "disappointed"}
```

[AnyHashable: Any]

# Any?



# Protocol Buffers

a.k.a protobuf

***Protocol buffers are Google's language-neutral,  
platform-neutral, extensible mechanism for  
serializing structured data – think XML, but smaller,  
faster, and simpler.***

**– Google**

***Serialization format***  
***- think JSON, but***  
***smaller, faster and safer***

— Yusuke

# Serialization Format

# Why are Protocol Buffers a big deal?

**HTTP Request**  
**JSON** ➔ **Data**

```
let json = ["id": 1]
let data = try? JSONSerialization.data(withJSONObject: json,
options: .prettyPrinted)

var request = URLRequest(url: url)
request.httpBody = data
```

# HTTP Request

# JSON ➔ Data

# **HTTP Request**

# **UserRequest → Data**

```
// custom type

let userRequest = UserRequest.with {
    $0.id = 1
}

let body = try? body.serializedData()

var request = URLRequest(url: url)
request.httpBody = body
```

# HTTP Response

Data → JSON

```
// {"user": {"name": "Yusuke"}}

URLSession.shared.dataTask(with: request) { (data, _, _) in
    let json = (try? JSONSerialization.jsonObject(with: data!, options: .allowFragments)) as? [AnyHashable: Any]
    let user = json?["user"] as? [AnyHashable: Any]
    let name = user?["name"] as? String
}
```

# HTTP Response

Data → JSON

# **HTTP Response**

# **Data → UserResponse**

```
// UserResponse(user: User(name: "Yusuke"))
URLSession.shared.dataTask(with: request) { (data, _, _) in
    // custom type
    guard let response: UserResponse = try? UserResponse(serializedData: data!) else {
        // error
        return
    }
    let user = response.user
    let name = user.name
}
```

```
Protobuf(  
    isTyped: true,  
    status: .satisfied  
)
```



**But how do  
they work?**



This repository

Search

Pull requests Issues Gist



apple / swift-protobuf

Watch ▾

45

Unstar

915

Fork

63

Code

Issues 31

Pull requests 2

Projects 0

Pulse

Graphs

Plugin and runtime library for using protobuf with Swift

# protobuf for Swift

thomasvl committed on GitHub Merge pull request #422 from thomasvl/more_ext_naming	Latest commit fc68a01 23 hours ago
Documentation	a month ago
Performance	29 days ago
Protos	23 hours ago
Reference	23 hours ago
Sources	23 hours ago
SwiftProtobuf.xcodeproj	a day ago
Tests	23 hours ago
.gitignore	a month ago
.jazzy.yaml	2 months ago
CollectTests.awk	a month ago

**"But we still use  
Objective-C 😢"**



This repository

Search

Pull requests Issues Gist



google / protobuf

Watch ▾

1,339

Unstar

16,046

Fork

4,794

Code

Issues 273

Pull requests 165

Projects 0

Wiki

Pulse

Graphs

Protocol Buffers - Google's data interchange format <https://developers.google.com/protocol-buffers/>

protobuf

protocol-buffers

protocol-compiler

protobuf-runtime

protoc

serialization

marshalling

rpc

4,507 commits

251 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

# protobuf

## google/protobuf

xfyjwf committed on GitHub Merge pull request #2897 from cerush/patch-5	Latest commit 373809e 3 days ago
benchmarks	6 months ago
cmake	9 days ago
conformance	16 days ago
csharp	8 days ago
docs	17 days ago
editors	8 months ago
examples	9 months ago
java	5 days ago
36 – Type-safe Web APIs with Protocol Buffers in Swift, Yusuke Kita (@kitasuke), iOSCon 2017	9 days ago
javanano	

1. Define message types
2. Generate code
3. Serialize/Deserialize

1. Define message types
2. Generate code
3. Serialize/Deserialize

**Message types define  
data structures  
in .proto files**

**Message types have key-value pairs**

# Type-safe API call with Protocol Buffers in Swift

Yusuke Kita

Apple recently open sourced swift-protobuf which is a plugin of Protocol Buffers for swift language. Protocol Buffers enables us to have type safety, make API faster and unify schema of structured data. JSON used to be a reasonable way for API call in most cases, but Protocol Buffers could be another option if we consider these benefits.

In this talk, you'll discover examples of usage with swift-protobuf in server and client apps. Yusuke will also highlight pros and cons compare to JSON based on his knowledge and experiences.

SWIFT api iOS json protocol-buffers ioscon type-safe

## About the speaker...



[Yusuke Kita](#)

Yusuke is an iOS developer at Mercari in San Francisco. He has been working on internationalization of Mercari app in iOS team. he is passionate about learning new technology in iOS. When not coding, you can find him cycling.

Yusuke tweets at [@kitasuke](#), and his blog can be found [here](#).

# user.proto

```
syntax = "proto3"; // protoc version
```

Type-safe API call with Protocol Buffers in Swift

Yusuke Kita

Apple recently open sourced swift-protobuf which is a plugin of Protocol Buffers for swift language. Protocol Buffers enables us to have type safety, make API faster and unify schema of structured data. JSON used to be a reasonable way for API call in most cases, but Protocol Buffers could be another option if we consider these benefits.

In this talk, you'll discover examples of usage with swift-protobuf in server and client apps. Yusuke will also highlight pros and cons compare to JSON based on his knowledge and experiences.

SWIFT api iOS json protocol-buffers loscon type-safe

## About the speaker...



Yusuke Kita

Yusuke is an iOS developer at Mercari in San Francisco. He has been working on internationalization of Mercari app in iOS team. he is passionate about learning new technology in iOS. When not coding, you can find him cycling.

Yusuke tweets at [@kitasuke](#), and his blog can be found [here](#).

```
message User {  
    int32 id = 1; // field number  
    string name = 2;  
    string introduction = 3;  
    string photoUrl = 4;  
    ParticipantType participantType = 5;  
  
enum ParticipantType {  
    Speaker = 0;  
    Attendee = 1;  
}  
}
```

# talk.proto

## Type-safe API call with Protocol Buffers in Swift

Yusuke Kita

Apple recently open sourced swift-protobuf which is a plugin of Protocol Buffers for swift language. Protocol Buffers enables us to have type safety, make API faster and unify schema of structured data. JSON used to be a reasonable way for API call in most cases, but Protocol Buffers could be another option if we consider these benefits.

In this talk, you'll discover examples of usage with swift-protobuf in server and client apps. Yusuke will also highlight pros and cons compare to JSON based on his knowledge and experiences.

SWIFT api iOS json protocol-buffers ioscon type-safe

### About the speaker...



Yusuke Kita

Yusuke is an iOS developer at Mercari in San Francisco. He has been working on internationalization of Mercari app in iOS team. he is passionate about learning new technology in iOS. When not coding, you can find him cycling.

Yusuke tweets at [@kitasuke](#), and his blog can be found [here](#).

```
syntax = "proto3";  
  
import "user.proto";  
  
message Talk {  
    int32 id = 1;  
    string title = 2;  
    string desc = 3;  
    User speaker = 4;  
    repeated string tags = 5; // Array  
}
```

1. Define message types
2. **Generate code**
3. Serialize/Deserialize

**protobuf compiler  
generates code  
from .proto file**

# Basic types

Int32, UInt32, Int64, UInt64, Bool, Float, Double, String,  
Array, Dictionary, Data

# Supported languages

C, C++, C#, Erlang, Go, Haskell, Java, Javascript,  
**Objective-C**, Perl, PHP, Python, Ruby, Rust, Scala,  
**Swift** etc.

# Swift features

- struct, not class
- enum RawValue is Int
- Default value is set



# user.proto

```
syntax = "proto3"; // protoc version
```

```
message User {  
    int32 id = 1; // field number  
    string name = 2;  
    string introduction = 3;  
    string photoUrl = 4;  
    ParticipantType participantType = 5;  
  
enum ParticipantType {  
    Speaker = 0;  
    Attendee = 1;  
}  
}
```

# user.pb.swift

```
// struct
struct User: SwiftProtobuf.Message, ... {
    init() {}

    enum Type: SwiftProtobuf.Enum {
        typealias RawValue = Int // always Int
        case speaker // = 0
        case participant // = 1
        case UNRECOGNIZED(Int)
    }

    // property has default value
    var id: Int32 = 0
    var name: String = ""
    var introduction: String = ""
    var photoURL: String = ""
    var participantType: User.ParticipantType = .speaker
}
```

# talk.proto

```
syntax = "proto3";  
  
import "user.proto";  
  
message Talk {  
    int32 id = 1;  
    string title = 2;  
    string desc = 3;  
    User speaker = 4;  
    repeated string tags = 5; // Array  
}
```

## talk.pb.swift

```
struct Talk: SwiftProtobuf.Message, ... {  
    init() {}  
  
    var id: Int32 = 0  
    var title: String = ""  
    var speaker: User = User()  
    var desc: String = ""  
    var tags: [String] = []  
}
```

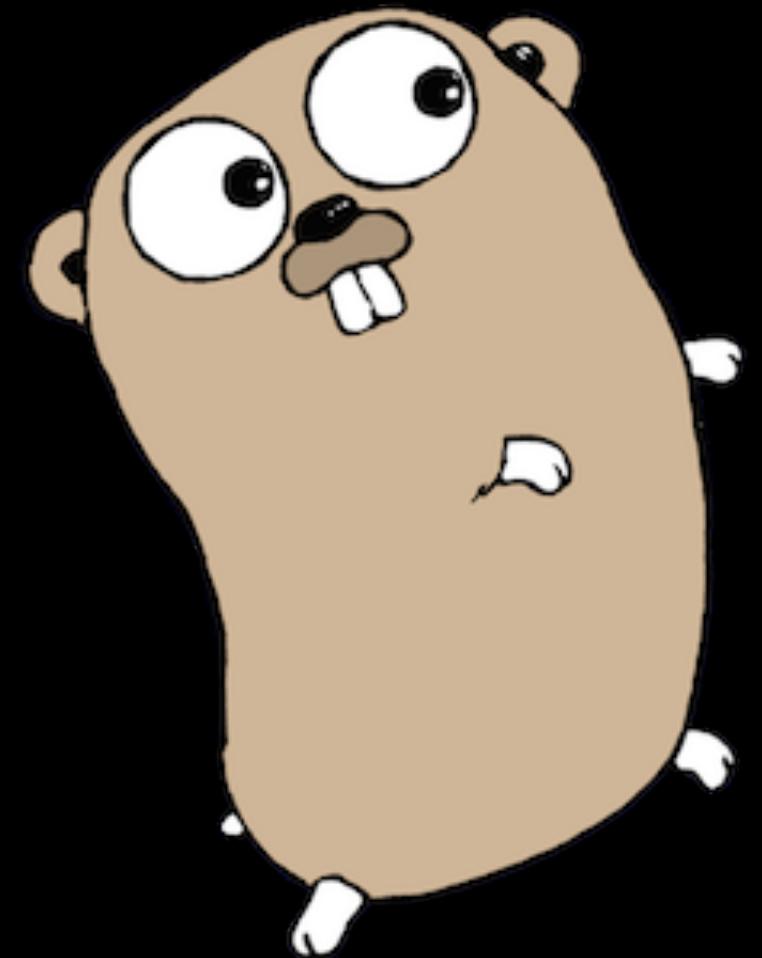


## User.java

```
public static final class User extends Message<User, User.Builder> {  
    public final Integer id;  
    public final String name;  
    public final String introduction;  
    public final String photoUrl;  
    public final ParticipantType participantType;  
}
```

## Talk.java

```
public final class Talk extends Message<Talk, Talk.Builder> {  
    public final Integer id;  
    public final String title;  
    public final User speaker;  
    public final String summary;  
    public final List<String> tags;  
}
```



# user.pb.go

```
type User_Type int32

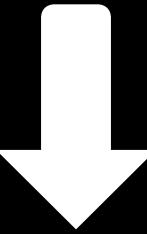
const (
    User_Speaker User_ParticipantType = 0
    User_Attendee User_ParticipantType = 1
)

type User struct {
    ID           int32  ^protobuf:"varint,1,opt,name=id,proto3" json:"id,omitempty" ^
    Name         string ^protobuf:"bytes,2,opt,name=name,proto3" json:"name,omitempty" ^
    Introduction string ^protobuf:"bytes,3,opt,name=introduction,proto3" json:"introduction,omitempty" ^
    PhotoURL    string ^protobuf:"bytes,4,opt,name=photoUrl,proto3" json:"photoUrl,omitempty" ^
    Type         User_Type ^protobuf:"varint,5,opt,name=type,proto3,enum=api.User_Type" json:"type,omitempty" ^
}
```

# talk.pb.go

```
type Talk struct {
    ID      int32      ^protobuf:"varint,1,opt,name=id,proto3" json:"id,omitempty"
    Title   string     ^protobuf:"bytes,2,opt,name=title,proto3" json:"title,omitempty"
    Speaker *User      ^protobuf:"bytes,3,opt,name=speaker" json:"speaker,omitempty"
    Summary string     ^protobuf:"bytes,4,opt,name=summary,proto3" json:"summary,omitempty"
    Tags    []string    ^protobuf:"bytes,5,rep,name=tags" json:"tags,omitempty"
}
```

# One message type



# Multiple languages code

**Less communication,  
More collaboration**

**with other platforms**

1. Define message types
2. Generate source files
3. **Serialize/Deserialize**

# Serialization

```
public func serializedData(partial: Bool = default) throws -> Data
```

```
public func jsonString() throws -> String
```

```
public func textFormatString() throws -> String
```

```
let user = User.with {  
    $0.id = 1  
    $0.type = .speaker  
    $0.name = "kitasuke"  
}  
  
let talk = Talk.with {  
    $0.id = 1  
    $0.title = "Type-safe Web APIs with Protocol Buffers in Swift"  
    $0.speaker = user  
    $0.tags = ["swift", "iOS", "protocol-buffers", "ioscon", "type-safe"]  
}  
  
let data = try? talk.serializedData()
```

# Deserialization

```
public convenience init(serializedData data: Data,  
extensions: SwiftProtobuf.ExtensionSet? = default,  
partial: Bool = default) throws  
  
public convenience init(jsonString: String) throws  
  
public convenience init(textFormatString: String,  
extensions: SwiftProtobuf.ExtensionSet? = default) throws
```

```
let talk = try? Talk(serializedData: data)
```

```
let title = talk?.title
```

```
let speaker = talk?.speaker
```

```
let tags = talk?.tags
```

# How serialization works

# Binary Encoding

# Key factor

1. Field number
2. Wire type

# Field number

```
message Talk {  
    int32 id = 1; < // Field number  
    string title = 2;  
    string desc = 3;  
    User speaker = 4;  
    repeated string tags = 5;  
}
```

# Wire type

Type	Meaning	Used For
0	Varint	int32, int64, uint32, uint64, sint32, sint64, bool, enum
1	64-bit	fixed64, sfixed64, double
2	Length-delimited	string, bytes, embedded messages, packed repeated fields
3		(deprecated)
4		(deprecated)
5	32-bit	fixed32, sfixed32, float

```
// message type  
message Test1 {  
    int32 a = 1;  
}
```

```
test1.a = 300
```

```
// encoded message
```

```
08 96 01
```

```
08 // field number and wire type
```

```
96 01 // value which is 300
```

# So small and numeric

# High network performance



This repository Search

Pull requests Issues Gist

Watch 0 Star 2 Fork 0

kitasuke / SwiftProtobufSample

Watch 0

Star 2

Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Pulse

Graphs

Settings

Sample project for client/server in Swift with Protocol Buffers

Edit

ios

swift

protobuf

kitura

swift-protobuf

protobuf-swift

Protocol

Buffers

Protocol

Buffers

Protocol

Buffers

Protocol

Buffers

Protocol

Buffers

Protocol

Buffers

25 commits

MIT

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

# Demo

## kitasuke/SwiftProtobufSample



kitasuke

1 commit · Last commit: f5e0082 20 days ago

Client

Tweaks

20 days ago

Server

Show data fetched by protobuf/json

a month ago

protos

Update example for talk

a month ago

.gitignore

Tweaks

3 months ago

LICENSE

Initial commit

3 months ago

README.md

Update README.md

a month ago

docs.md

75 — Type-safe Web APIs with Protocol Buffers in Swift, Yusuke Kita (@kitasuke), iOSCon 2017

3 months ago

# Quick Recap

# Quick Recap

→ Type-safety

# Quick Recap

- Type-safety
- Shared data model

# Quick Recap

- Type-safety
- Shared data model
- High performance

You might have concerns  
about...

# versioning

# Backward compatibility

- Unknown field is ignored
- Default value for missing field

# Backward compatibility

- Unknown field is ignored
- Default value for missing field

**as long as you don't change existing field number or wire type**

# Coexistence of **protobuf & JSON**

# Absolutely you can!

# Accept & Content-Type

- application/protobuf - protobuf
- application/json - JSON

# HTTP Request

```
var request = URLRequest(url: url)
if protobuf {
    request.setValue("application/protobuf", forHTTPHeaderField: "Content-Type")
    request.setValue("application/protobuf", forHTTPHeaderField: "Accept")
} else if json {
    request.setValue("application/json", forHTTPHeaderField: "Content-Type")
    request.setValue("application/json", forHTTPHeaderField: "Accept")
}
```

# HTTP Response

```
URLSession.shared.dataTask(with: request) { (data, urlResponse, _) in
    let httpURLResponse = urlResponse as? HTTPURLResponse
    let contentType = httpURLResponse?.allHeaderFields["Content-Type"] as? String

    if contentType == "application/protobuf" {
        let response = try? Response(serializedData: data!)
    } else if contentType == "application/json" {
        let response = try? Response(jsonString: data!)
    }
}
```

Sounds good  
so far



**So, what are  
cons?**

# Not human-readable

→ Binary data is not understandable

# Time investment

- Time consuming at the beginning
- Involvement from other platforms

# Swift version

- Watch Swift version of protobuf plugin
- Specify tag version if you use older version

# Stability

- Still pre-release version only for Swift ⚡
- Contribute if you find a bug 

**Where should  
we start?**

# Internal service

- Easy to adapt
- Small start

# Data store

→ Database

→ NSKeyedArchiver

→ NSCache

# Parsing library

- JSON parser
- Server side Swift

# Conclusion

# Conclusion

→ Swifty

# Conclusion

- Swifty
- Consistent in Cross-platform

# Conclusion

- Swifty
- Consistent in Cross-platform
- Extensible

**It's definitely  
worth it** 

# Partially in Production 70M Downloads



# Credits

Protocol Buffers  
swift-protobuf  
Kitura

Protocol Buffers in your Kitura Apps

# Thank you!

GitHub: <https://github.com/kitasuke>

Demo: [SwiftProtobufSample](#)

Twitter: [@kitasuke](#)

Email: [yusuke2759@gmail.com](mailto:yusuke2759@gmail.com)